UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en ingeniería informática

TRABAJO FIN DE GRADO

Aplicación para la Gestión Interna de Eventos

Álvaro Muñoz Sanz-Gadea
Tutor: Daniel Hernández Lobato

JUNIO 2016

Aplicación para la Gestión Interna de Eventos

AUTOR: Álvaro Muñoz Sanz-Gadea
TUTOR: Daniel Hernández Lobato

Dpto. Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Junio de 2016

Resumen

En este Trabajo Fin de Grado se ha llevado a cabo un proyecto de software en el que se ha desarrollado una aplicación web para la gestión interna de grandes eventos como ferias, conciertos o encuentros deportivos.

Tras trabajar durante varios años en varias empresas dedicadas al sector de la organización de eventos y protocolo, he podido detectar una serie de problemas comunes en estas empresas. Este proyecto se centra en solucionar estos problemas y ayudar a coordinar y gestionar mejor este tipo de organizaciones, que hasta ahora han tenido una línea muy tradicional.

Para la realización de este trabajo se ha seguido el ciclo de vida de un proyecto de software en el que se ha analizado el problema y se han sacado una serie de conclusiones sobre lo que había que mejorar. Estas conclusiones, son los requisitos con los que partía el proyecto y a partir de los cuales se ha seguido buscando mejoras para una gestión más eficiente durante los eventos. En el segundo paso, se ha analizado el panorama tecnológico actual con la idea de ver cuál era el entorno de desarrollo que mejor se ajustaba a las necesidades. Como se verá con mayor profundidad, el lenguaje que finalmente se ha decidido usar, ha sido Java. Como tercer fase se ha diseñado la aplicación, tanto la lógica como la presentación.

Una vez se ha analizado el problema y se ha diseñado una solución, se ha trabajado en su desarrollo y pruebas para lograr una aplicación que contara con todas las funcionalidades que se habían descrito.

Finalmente, en este documento, se ha descrito todo este trabajo en el que se cuenta en profundidad cada uno de los pasos llevados a cabo y las justificaciones oportunas para explicar las decisiones tomadas.

Palabras clave

Aplicación, Beans, Controlador, Framework, Java, JavaServer Faces, Modelo, PrimeFaces, Vista, Web.

Abstract

In this project, a web application for internal organization purpose of different events such as concerts or sports events was developed.

After working several years in different companies dedicated to the sector of events organization and protocol, I have been able to identify common issues in this type of companies. The main aim of this project is to focus in solving these issues as well as helping to coordinate in a more efficient way this type of organizations which are mostly traditional companies.

For the development of this project, it has been developed a software project in which the problem was analyzed in order to obtaining some conclusions on how to solve these key issues. These conclusions have been the starting points of the project and from which improvements have been seek in order to obtain a more efficient management during the events.

The current technological environment has been analyzed for the purpose of identifying the development environment that best suits the necessity of this type of organizations. As it will be seen with greater depth, Java is the chosen language to develop this project. To continue, has been designed the application, from the logic to the views.

Once the different problems were identified and the solutions were designed, an application with all the features to meet the requirements in order to solve the problems was designed.

Finally, in this document, all this work was described in more depth, identifying each issues and the opportunity justification for which there were made.

Keywords

Application, Beans, Controller, Framework, Java, JavaServer Faces, Model, PrimeFaces, View, Web.

Agradecimientos

A ti, por interesarte.

INDICE DE CONTENIDOS

INDICE DE FIGURAS

1 Introducción

1.1 Motivación

Desde hace ya unos años las aplicaciones web son parte de nuestro día a día. Tareas tan comunes como consultar las cuentas del banco o pedir hora para renovar el DNI se hacen a través de estas aplicaciones. Desde que en diciembre de 1990 el inglés Tim Berners-Lee acabara el proyecto de la "World Wide Web" durante su estancia en el CERN, su evolución ha sido exponencial suponiendo una revolución global en la presentación de contenidos y la gestión a través de ellos.

En concreto, existe un problema en la organización de grandes eventos como ferias, conciertos o encuentros deportivos que requieren de un gran número de trabajadores. En estos eventos se coordinan diferentes empresas independientes como pueden ser las que llevan el catering, la seguridad o el protocolo. La organización intrínseca de estas empresas se basa en dos principios:

- El uso limitado de la tecnología para su gestión debido a un método organizativo tradicional.

- La escasa inversión en software. Esto viene motivado porque las grandes empresas suelen subcontratan a otras por un período corto de tiempo, provocando una gran rotación en este sector.

Cabe destacar la casuística de las empresas de azafatas. Dichas empresas suelen ser pequeñas organizaciones que no pueden permitirse un gran desembolso en la inversión de software motivado por la incertidumbre por la renovación de su contrato a medio plazo.

Tras haber trabajado durante cuatro años en una empresa de organización de eventos de un equipo de futbol de gran importancia en el ámbito nacional, he podido detectar deficiencias existentes en la organización. Esto me ha motivado a utilizar mis conocimientos informáticos para mejorar la situación de estas empresas de cara a realizar una gestión más eficiente de su tiempo y sus recursos.

1.2 Objetivos

Con la intención de mejorar la gestión de este tipo de eventos en los que su coordinación sigue siendo bastante tradicional se ha planteado este proyecto.

Se han identificado diferentes problemas comunes a las empresas dedicadas al sector de la organización de eventos. El objetivo de este proyecto es reducir dichos problemas. A continuación se detallan los mismos:

1. Envío de correos electrónicos, previos al evento, a los empleados de manera manual provocando pérdida de tiempo y entrando en juego el factor del error humano. Con la herramienta que se presenta, los emails se enviarían de manera automática de forma que se ahorraría tiempo y se mitigaría, de esta manera, el riesgo de error humano.

2. Déficit en la gestión de los eventos futuros. Normalmente, estas empresas avisan a sus empleados con un plazo máximo de dos semanas previas al evento. Gracias a la herramienta desarrollada, se podrá consultar un calendario anual con los próximos eventos.

3. Gestión del pago de nóminas a los empleados. La herramienta genera un informe en el que se detalla los eventos trabajados por empleado y el sueldo de los mismos. De esta manera, se puede consultar el importe a pagar, cada mes, a cada empleado. Esto facilitaría y ahorraría tiempo a los organizadores de los eventos. Además, los empleados pueden consultar el desglose de los eventos realizados por los que reciben una retribución.

4. Errores en la comunicación, durante los eventos, derivados del uso de herramientas arcaicas para dicho fin como por ejemplos talkie walkie. Con este sistema, los organizadores pueden consultar los empleados que se encuentran trabajando y los puestos designados a cada uno, reduciendo las dudas que puedan surgir durante los eventos.

5. Posible falta de confidencialidad en los datos de los empleados. Las empresas de organización de eventos cuentan con un gran número de datos personal de cada trabajador. El proyecto se ha desarrollado teniendo en cuenta la importancia de la seguridad de la información que se guarda.

Por último, la herramienta facilitará la organización ya que se dispondrá de una ficha detallada de cada empleado así como de una galería de fotos de los mismos. Con esta información, podrán presentar las fichas de los empleados a futuros de clientes facilitando así la satisfacción del cliente con los azafatos.

1.3 Organización de la memoria

La memoria consta de los siguientes capítulos:

- Capítulo 2: **Estado del arte**. Estudio sobre las tecnologías actuales más relevantes en el desarrollo de aplicaciones web y la explicación de la tecnología usada dentro de este marco tecnológico.
- Capítulo 3: **Análisis y Diseño**. Informe de requisitos funcionales y no funcionales con los que debe contar la aplicación. Ciclo de vida del proyecto y organización de los hitos.
- Capítulo 4: **Desarrollo**. Explicación de los hitos llevados a cabo.
- Capítulo 5: **Integración, pruebas y resultados**. Migración al entorno de producción y pruebas realizadas durante ciclo de vida del proyecto.
- Capítulo 6: **Conclusiones y trabajo futuro**. Conclusiones finales sobre el proyecto y explicación de los próximos pasos.

2 Estado del arte

2.1 Frameworks de desarrollo

En este capítulo se expone un resumen de la investigación realizada a priori del proyecto con el objetivo de ver que tecnología era la más adecuada para su desarrollo. Esta búsqueda de información consistió en buscar los frameworks actualmente usados para ver sus ventajas e inconvenientes. El objetivo no es sacar una conclusión sobre cuál es mejor o peor, tan solo justificar mediante las caracterísiticsas de cada una de las tecnologías de desarrollo web, la elección tomada para la elaboración del proyecto.

2.1.1 Django

Es un framework de desarrollo web para Python. Su mayor característica es la simplicidad, el desarrollo rápido y centrado en la máxima de no repetir código para reusar lo ya programado. Está basado en el patrón que se verá en más ocasiones Modelo-Vista-Controlador (MVC). Aunque cuenta con una pequeña inconsistencia en su nomenclatura, ya que el controlador es llamado vista y se refiere a las vistas como templates. Este framework trae su propio servidor web y con la base de datos SQLite.

Django provee tres puntos diferentes en los que permite ejecutar clases middleware, previamente definidas en el archivo de configuración:

Request middleware: Se ejecuta tras crear el objeto HttpRequest y justo antes de resolver la URL, permitiendo modificar el objeto request o devolver una respuesta propia antes de que el resto de la aplicación ejecute.

View middleware: Es ejecutado después de la resolución de la URL, pero antes de ejecutar la vista correspondiente. Esto permite ejecutar operaciones antes y después de la ejecución de la vista. La vista podría llegar a no ejecutarse en absoluto.

Response middleware: Se ejecuta al final, después de que el objeto response haya sido creado y antes de entregarlo al cliente. Es utilizado para realizar las modificaciones finales.

2.1.2 JavaServer Faces

JSF es un framework para desarrollo web en Java. Como Django, está basado en el patrón MVC.

El modelo JSF se define mediante “Beans”, que son clases java con un conjunto de atributos y métodos. Estos atributos se denominan propiedades y pueden ser leídas o escritas desde las páginas JSF. El modelo es la encapsulación de los datos.

La vista es la presentación del modelo. La forma más común de presentar la información es a través de ficheros XHTML con etiquetas propias que definen componentes JSF.

El controlador define la lógica interna. Reacciona a la petición del cliente.

La comunicación entre vista y controlador se hace por medio de los “Managed Beans”. Los cuales se les da valor a través de la vista para ser usados en el controlador o la lógica les proporcionan un valor para que sean mostrados a través de la vista.

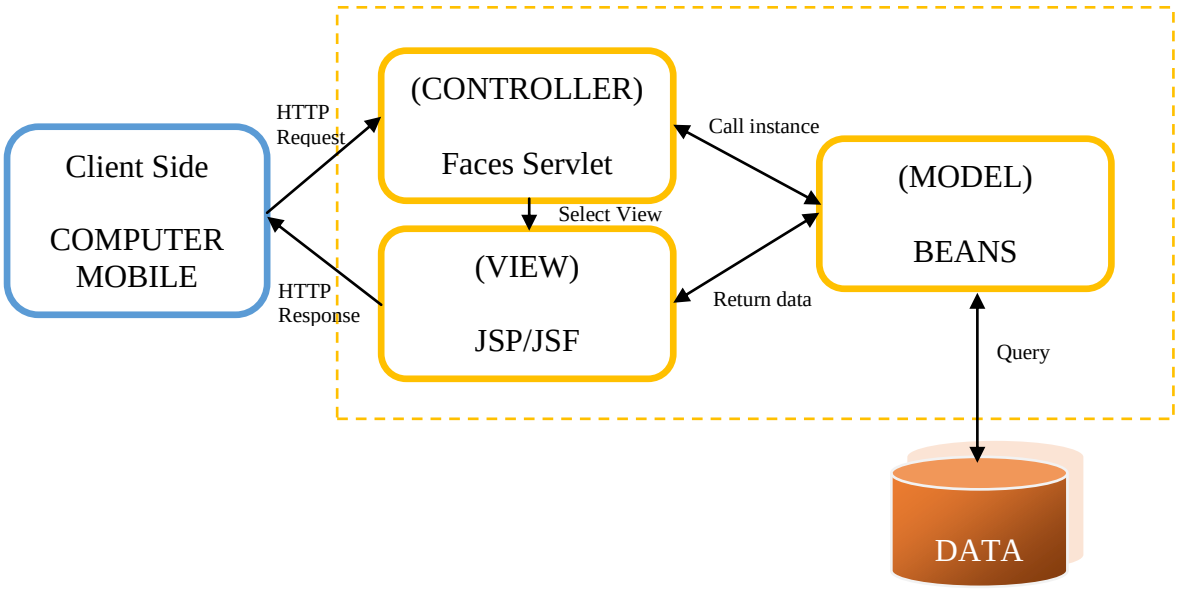


Figura 2.1.2-1: JavaServer Faces

2.1.3 Symfony

Framework para desarrollo en php. Está escrito en la versión php5 y es la que se recomienda. Al igual que las tecnologías anteriores usa un patrón MVC y es compatible con la mayoría de gestores de bases de datos. La presentación o vista se desarrolla en HTML lo que facilita su uso sin conocimiento del framework.

Ente sus características cabe destacar su facilidad a la hora de instalarlo y configurarlo. Buen rendimiento y bajo consumo de la memoria.

2.1.4 Ruby on Rails

Este Framework de desarrollo web vuelve a ser de código abierto y usa el patrón MVC. Está escrito en Ruby. Intenta simplificar al máximo escribiendo el menor código posible. Su principio fundamental es "No te repitas". El desarrollador tendrá que hacer el mínimo de configuración ya que sólo necesita definir aquella configuración que no es convencional.

Aunque es una de las tecnologías más de moda actualmente, todavía no cuenta con demasiado apoyo de la comunidad de desarrolladores ni se puede encontrar tanta documentación como sobre otras tecnologías.

2.1.5 ASP.NET

Framework propietario de Microsoft. Está construido sobre el Common Language Runtime, permitiendo a los programadores escribir código ASP.NET usando cualquier lenguaje admitido por el .NET Framework.

ASP.NET soporta tres modelos diferentes de desarrollo: "Web Pages", MVC y "Web Forms".

2.2 Gestores de bases de datos

En el proyecto, como en la mayoría de las aplicaciones web, se va a contar con una base de datos donde almacenar la información. Casi la totalidad de estos datos son recogidos a través de formularios web como pueden ser los datos de las personas registradas y que trabajan en la agencia, la información de los eventos,…

2.2.1 MongoDB

Esta aplicación es diferente a las demás que se presentan. En vez de seguir la línea clásica es un sistema NoSQL. En lugar de guardar los datos en tablas como se hace en las base de datos relacionales, guarda estructuras de datos en documentos BSON con un esquema dinámico, esto permite que la integración de los datos en ciertas aplicaciones sea más fácil y rápida.

2.2.2 MySQL

Gestor de base de datos relacional, posiblemente la más usada a nivel mundial. Cuenta con disponibilidad en gran cantidad de plataformas y sistemas. Se centra en la optimización de las consultas. Su gran popularidad reside en aplicaciones web desarrolladas en PHP. Aunque es software libre, Oracle es dueña de la licencia.

2.2.3 Oracle Database

Sistema de gestión de base de datos de tipo objeto-relacional (ORDBMS), desarrollado por Oracle Corporation. Es considerado como uno de los sistemas de bases de datos más completos, destacando: soporte de transacciones, estabilidad, escalabilidad, y soporte multiplataforma.

2.2.4 PostgreSQL

Sistema de gestión de bases de datos relacional orientado a objetos y libre. En este caso, no existe una compañía dueña de las licencias y es la propia comunidad de desarrolladores la que se encarga de continuar con su evolución. Entre sus características destaca la alta concurrencia, permitiendo que usuarios accedan en modo lectura a una tabla mientras está siendo editada por otro usuario, sin bloquearlo.

2.2.5 SQL Server

También conocido como Microsoft SQL Server ya que es sistema de manejo de bases de datos del modelo relacional, desarrollado por la empresa Microsoft. SQL Server solo está disponible para sistemas operativos Windows de Microsoft.

2.2.6 SQLite

Es un sistema de gestión de bases de datos relacional contenida en una biblioteca escrita en C. A diferencia de los sistemas anteriores con un modelo cliente-servidor, el motor de SQLite no es un proceso independiente, la biblioteca SQLite se enlaza con el programa principal. Este utiliza la funcionalidad de SQLite a través de llamadas simples a subrutinas y funciones. Esto ayuda a reducir la latencia en el acceso a la base de datos, debido a que las llamadas a funciones son más eficientes que la comunicación entre procesos. Aunque en su versión 3 (SQLite3) permite bases de datos de hasta 2 Terabytes de tamaño está recomendado para bases de datos pequeñas.

2.3 Conclusiones

Ya que uno de los hitos futuros de este proyecto es la integración con aplicaciones para móviles Android y el desarrollo de estas aplicaciones se hace en Java, se vió que la mejor opción por temas de integración era unificar tecnologías por lo que se ha decidido que el desarrollo de la aplicación web también sea en Java. Finalmente se ha decido usar el framework JavaServer Faces.

Una virtud de JSF es el poder hacer uso de la API Enterprise Java Beans (EJB) que permite al desarrollador centrarse en la lógica de negocio y abstraerse de problemas generales como concurrencia, transacciones, persistencia, seguridad, etc. Gracias a que los EJB están basados en componentes y que pueden soportar invocación remota permite una gran flexibilidad y pueden ser reutilizado por las aplicaciones para Android.

JavaServer Faces usa la tecnología XHTML para las vistas de la aplicación. Esta tecnología encapsula el código HTML en etiquetas XML. Una ventaja de esta tecnología es que permite integrar AJAX de manera muy fácil. Esto nos permite que la aplicación se ejecute en el lado del cliente mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible presentar vistas dinámicas en las que se realizan cambios sobre las páginas sin necesidad de recargarlas, mejorando así la interactividad, velocidad y usabilidad de la aplicación.

Una de las librerías de JSF es PrimeFaces. Esta librería de código abierto cuenta con un conjunto de componentes enriquecidos que facilitan la creación de las aplicaciones web. Sus componentes cuentan con un diseño innovador que ayuda a crear interfaces amigables para el usuario.

A continuación se puede ver la tendencia de popularidad según Google de la librería PrimeFaces en los últimos años:

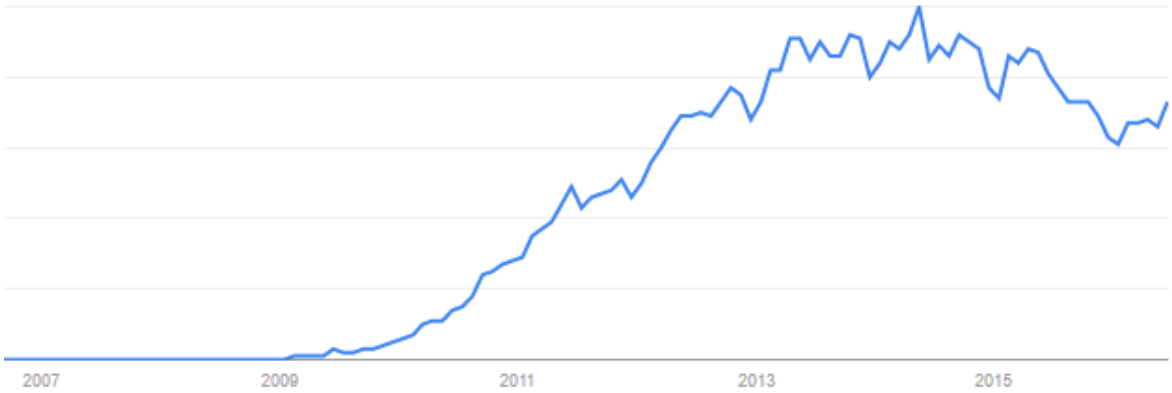


Figura 2.3-1: Índice de popularidad de PrimeFaces

Se ha decidido usar el entorno Netbeans para el desarrollo del proyecto. Este entorno provee los drivers para varios servidores de bases de datos y así poder conectarse de manera sencilla con estos sistemas.

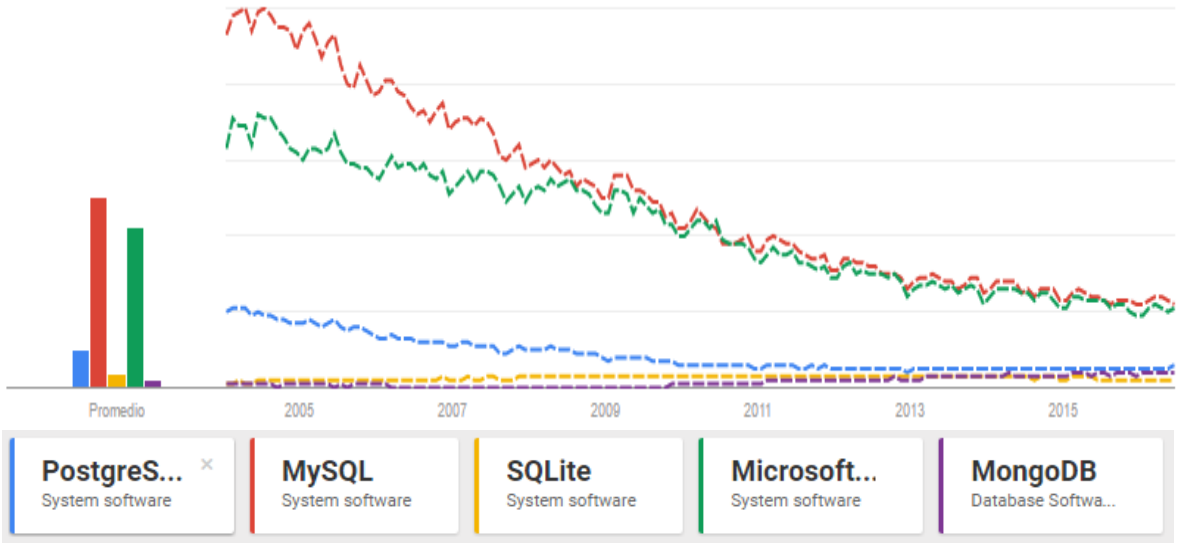


Figura 2.3-2: Índice de popularidad de Gestores de Bases de Datos

En el diagrama se puede ver cómo ha ido evolucionando la popularidad de los gestores más usados según Google. Los dos sistemas con más adeptos son MySQL y Microsoft SQL Server seguidos de PostgreSQL.

Microsoft SQL Server está diseñado para usarse en el sistema operativo Microsoft Windows. Nuestro entorno de desarrollo será en Ubuntu por lo que son mejor opción MySQL o PostgreSQL, dos gestores que tienen un buen rendimiento en Linux.

Tanto MySQL como PostgreSQL son de código abierto. Sin embargo, MySQL está distribuido bajo una licencia "Copyleft" la cual obliga a los vendedores de software propietario a liberar su código o adquirir una licencia propietaria de Oracle. Sin embargo, PostgreSQL está distribuido términos más permisivos.

PostgreSQL destaca por su fiabilidad e integridad de los datos. Cuenta con un planificador de consultas extremadamente sofisticado. MySQL se centra en la optimización de consultas sencillas, tradicionalmente se ha utilizado en aplicaciones web escritas en PHP.

Según las características que presentan los diferentes gestores se ha decido que el que mejor se ajusta a las necesidades de este proyecto es PostgreSQL y es el que se ha usado para la base de datos.

3 Análisis

3.1 Introducción

3.1.1 Propósito del sistema

El fin del proyecto es mejorar la organización y gestión de grandes eventos por medio de una aplicación web.

3.1.2 Ámbito del sistema

El sistema debe permitir a las personas poder registrarse en la agencia de manera online y guardar esta información en la base de datos. Los organizadores podrán crear, modificar o cancelar eventos. En función de las características del evento, el sistema enviará convocatorias a los empleados que deberán confirmar su asistencia.

Durante el transcurso del evento cualquier coordinador podrá consultar la lista de los empleados y sus puestos de trabajo. Cualquier modificación la podrán ver en tiempo real el resto de coordinadores, como puede ser la baja de un empleado o la sustitución de una persona por otra.

En cualquier momento un empleado puede consultar el calendario con los próximos eventos organizados para así saber con mayor anticipación que días va a ser convocado para trabajar. Esto reducirá el número de imprevistos por falta de organización de los empleados.

3.1.3 Objetivo

Los objetivos básicos de la aplicación es el de poder registrar la información de los empleados y de los eventos en el sistema. La información (con restricciones según el perfil del usuario) tiene que estar disponible en todo momento para poder ser consultada. Esto ayudará tanto a los organizadores del evento como a los empleados a gestionarse mejor y reducir el número de imprevistos.

3.1.4 Definiciones

Azafatos: Empleados con las funciones de imagen y protocolo en las zonas VIP del evento.

Coordinadores: Los jefes de equipo. Son los responsables del correcto funcionamiento de la zona donde están asignados. Tienen permiso en el sistema para consultar información y realizar alguna modificación restringido a su zona.

Convocatoria: Citación del empleado para trabajar en un evento.

Empleados: Toda persona que trabaja en el evento.

Empleados de catering: Personas encargadas de la comida.

Empleados de limpieza: Personas encargadas de la higiene y presentación del recinto.

Empleados de seguridad: Personas con la función de mantener el orden durante el evento.

Equipos: Grupo de empleados de una zona.

Organizadores: Las personas encargadas de definir las características de un evento. Tienen permiso para crear, modificar o cancelar eventos.

Palco: Sinónimo de zona VIP en algunos recintos como los deportivos o teatros.

Recintos: Lugar donde tienen lugar los eventos.

Suplente: Empleado sin puesto de trabajo asignado que asiste al evento para suplir cualquier baja que pudiera ocurrir en el momento de comenzar el evento.

Zona: Los recintos donde tienen lugar los eventos se dividen por zonas para facilitar su organización.

Zona VIP: Zonas exclusivas con el acceso restringido a unas entradas especiales.

3.2 Descripción del sistema

3.2.1 Requisitos funcionales

RF1 El administrador del sistema será el encargado de introducir en la base de datos a los organizadores.

RF2 Los organizadores pueden crear, modificar o cancelar eventos.

RF3 Los organizadores podrán dar privilegios de coordinador a los empleados.

RF4 Los coordinadores tendrán permiso de lectura para todo el sistema y restringido a su zona para escritura.

RF5 Los coordinadores podrán dar de baja a alguno de los miembros de su equipo.

RF6 Durante el evento el coordinador que tenga una baja podrá elegir entre los suplentes disponibles quien ocupará ese puesto de trabajo.

RF7 Los empleados podrán acceder a cierta información como la del evento al que han sido convocados.

RF8 Los empleados que han sido convocados para un evento deberán su asistencia en las siguientes 24 horas.

RF9 El empleado que habiendo sido convocado para un evento y no vaya a poder asistir deberá informar al sistema de su baja. En caso de no hacerlo su coordinador será el que informe de la baja del empleado para poder elegir un suplente.

RF10 Cualquier persona que quiera trabajar en la agencia tendrá que registrarse en el sistema desde la plataforma web.

3.2.2 Requisitos no funcionales

RNF1 El sistema debe funcionar correctamente en diferentes entornos (Diferentes navegadores y sistemas operativos).

RNF2 Los organizadores podrán crear tantos eventos como desee.

RNF3 La información debe ser persistente y accesible en todo momento.

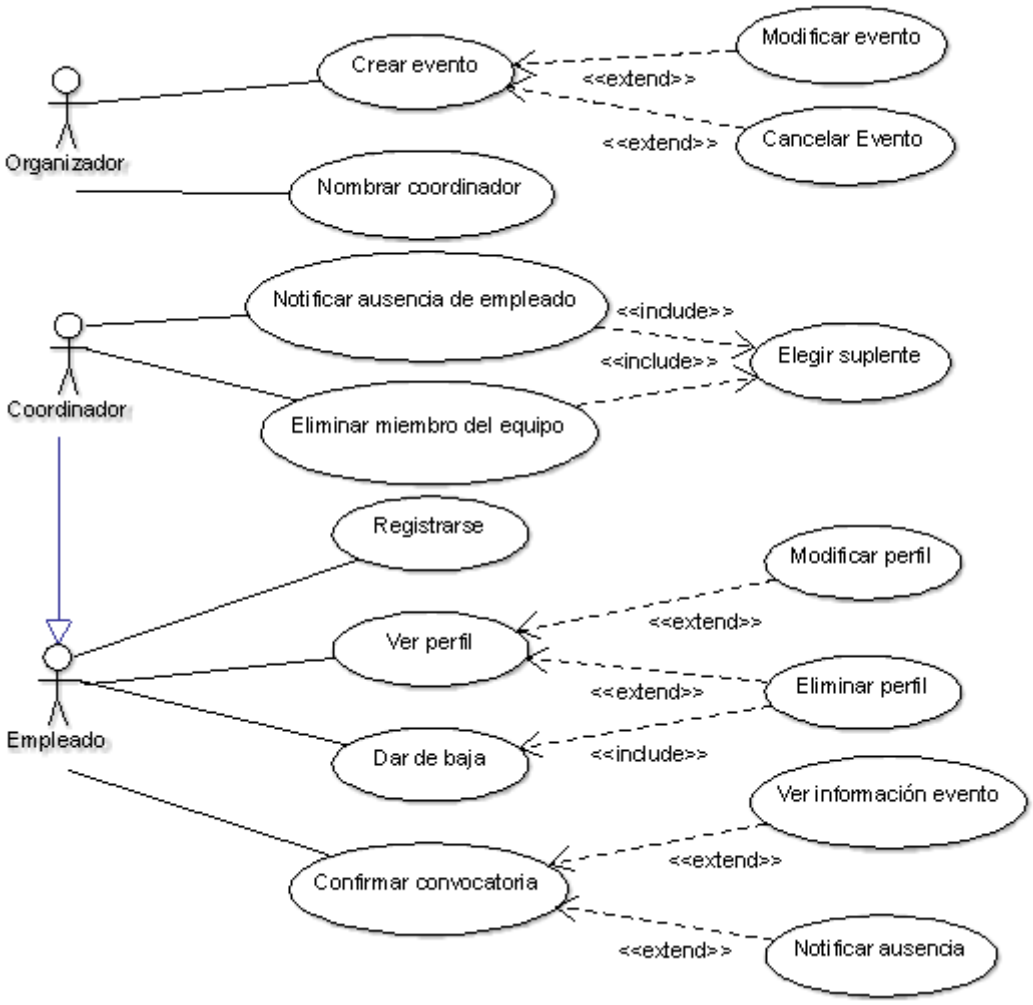
RNF4 El rendimiento no es un punto crítico del sistema.

RNF5 La interfaz debe ser lo más sencilla posible y tener una buena presentación.

RNF6 Debe existir un backup actualizado del sistema que pudiera remplazar el original en caso de fallo.

4 Diseño

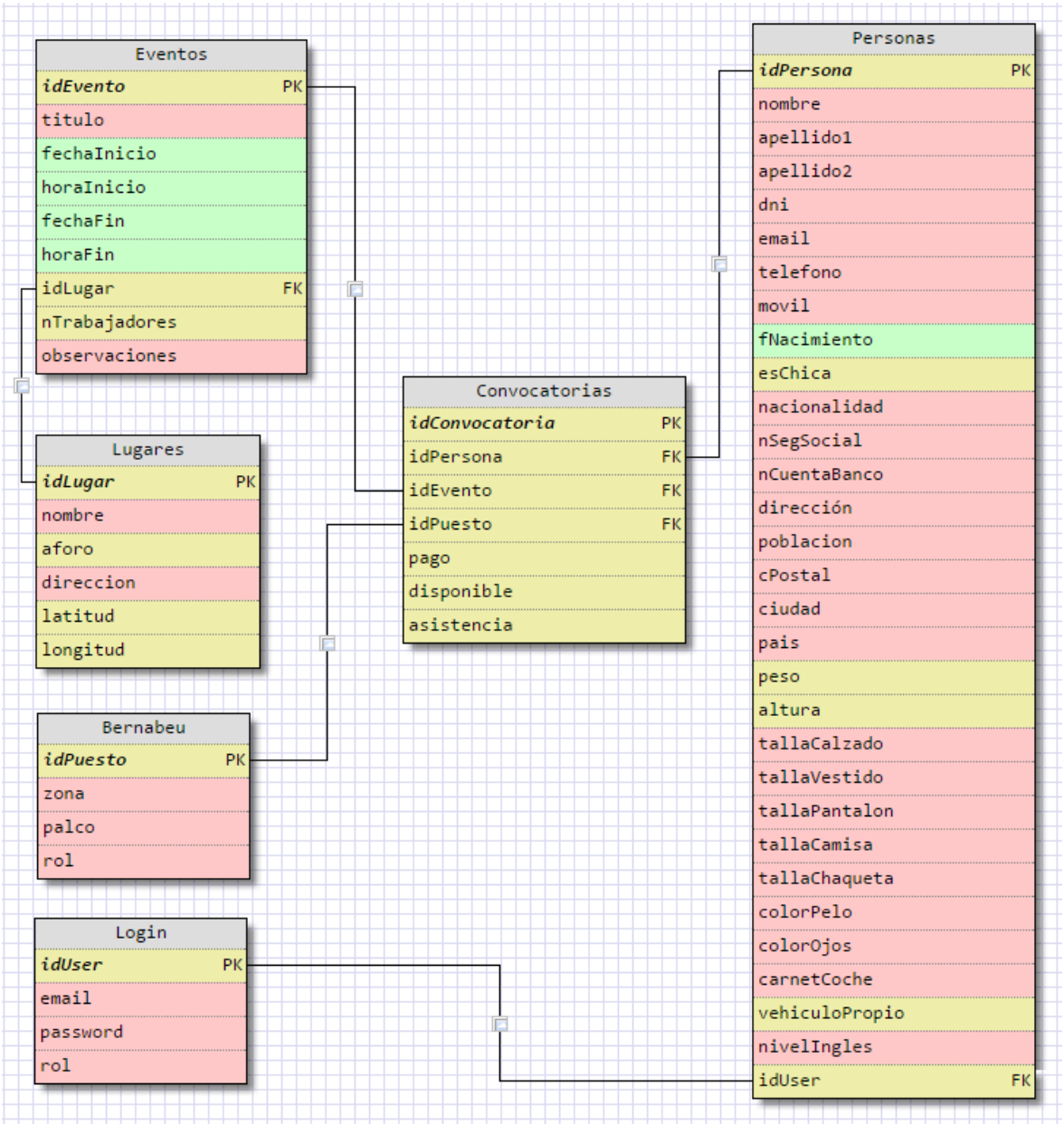
4.1 Casos de Uso



4.2 Esquema relacional de la base de datos

El esquema de la base de datos consta de 5 tablas básicas (Login, Personas, Lugares, Eventos y convocatorias).

Se ha presentado una sexta tabla a modo de ejemplo de donde se recoger la información sobre cómo está estructurado un recinto. En la práctica existirá una tabla para cada recinto donde se organiza un evento (Estadio Santiago Bernabéu, IFEMA, Estadio Vicente Calderón, Palacio de los Deportes,…)



4.3 Diagrama de secuencia

A continuación se van a representar tres diagramas de secuencia para diferentes casos de envíos de convocatorias.

El caso base sería el siguiente: El organizador crea un evento en el sistema, este envía una pre-convocatoria a un empleado. Esta persona confirma que tiene disponibilidad para trabajar ese día y que se puede contar con él. Entonces se le envía una convocatoria citándole para el evento y él lo confirma para saber que lo ha recibido y leído.

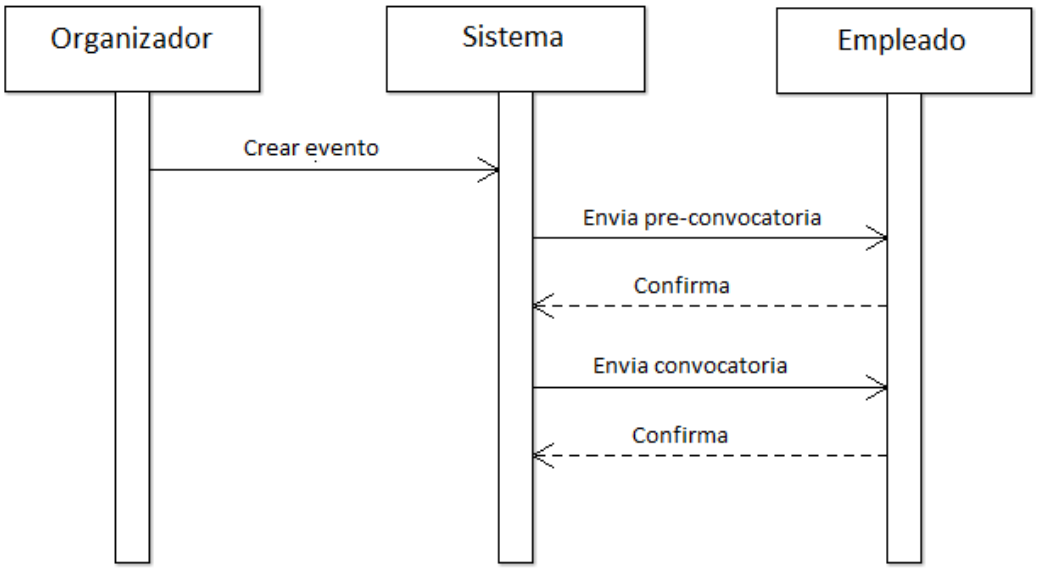


Figura 4.3-1: Diagrama de secuencia caso base.

No siempre se cumple el caso base por lo que hay que contar un equipo de suplentes que estén disponibles para trabajar ese día en caso de que alguien falle. En el segundo diagrama representamos una situación así.

El sistema envía una pre-convocatoria al empleado. Este confirma su disponibilidad. Cuando el sistema recibe todas las confirmaciones, comprueba que no tiene un margen suficiente de suplentes disponibles para el caso de que falle gente del equipo.

Envía la convocatoria final citando al empleado que había confirmado la pre-convocatoria para el evento. El sistema recibe la confirmación a la citación del empleado y la confirmación de disponibilidad del suplente por si al final se le necesita.

Por causas personales el empleado notifica que no va a poder asistir al evento. En ese momento el sistema envía una convocatoria al suplente citándole para el evento con el puesto que se ha quedado libre.

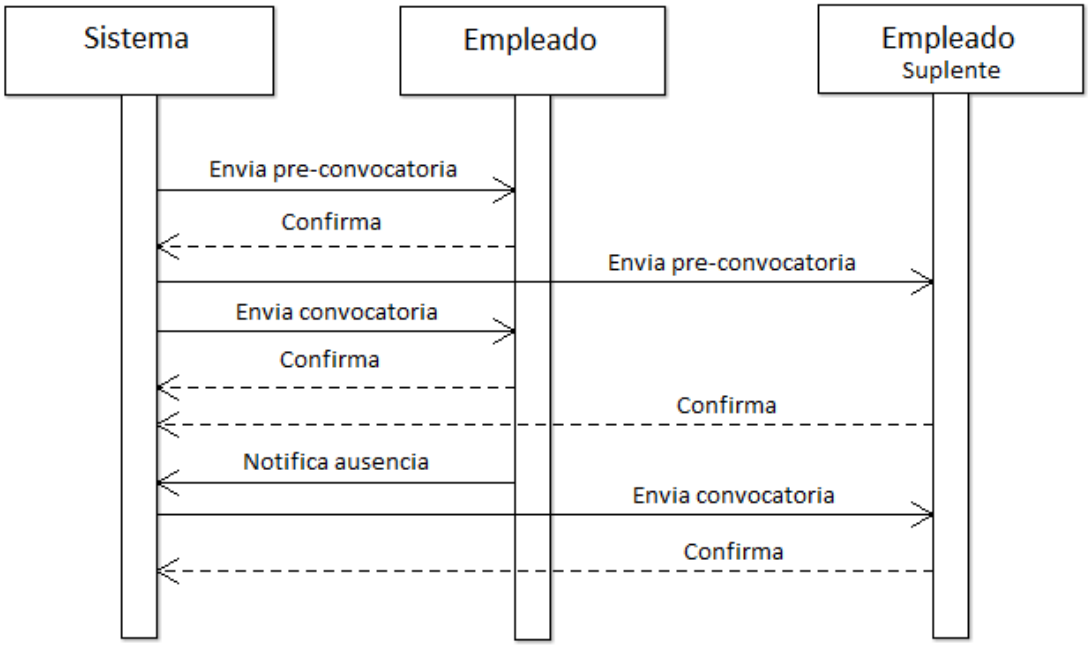


Figura 4.3-2: Diagrama de secuencia con suplente.

En el tercer caso se contempla la situación en la que el empleado que había confirmado la asistencia al evento no se presenta en su puesto de trabajo. Hay un grupo de persona que van al evento como suplentes (sin puesto asignado) para cubrir faltas de última hora. Si finalmente no tienen que cubrir ninguna baja se quedan reforzando alguna zona. En el momento de la citación el coordinador ve que le falta alguna persona de su equipo notifica la ausencia de esa persona, puede ver la lista de suplentes disponibles y elige uno de ellos.

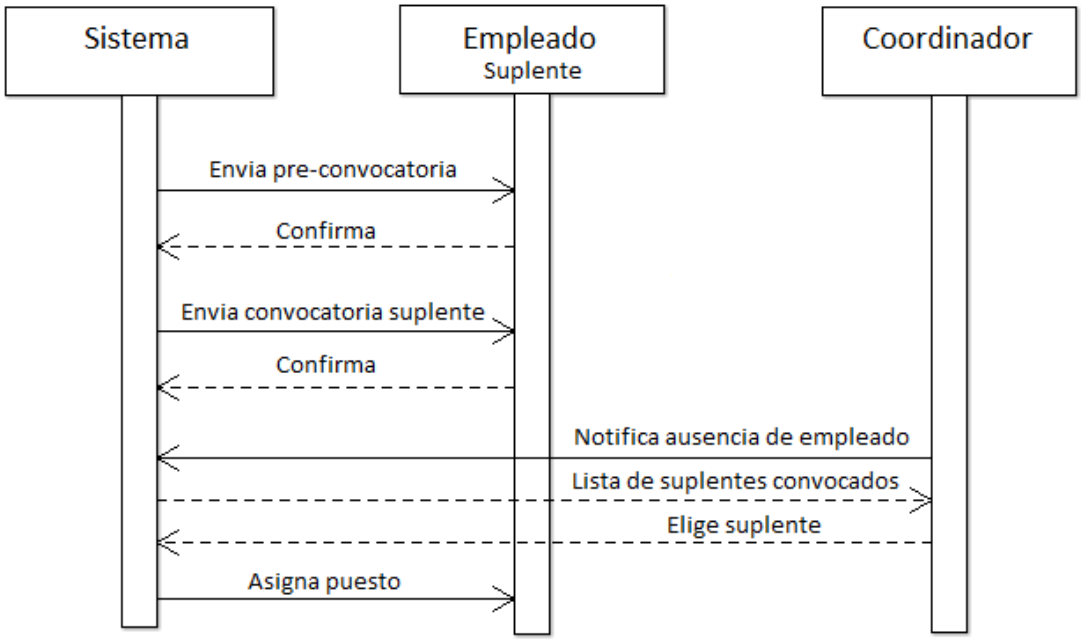
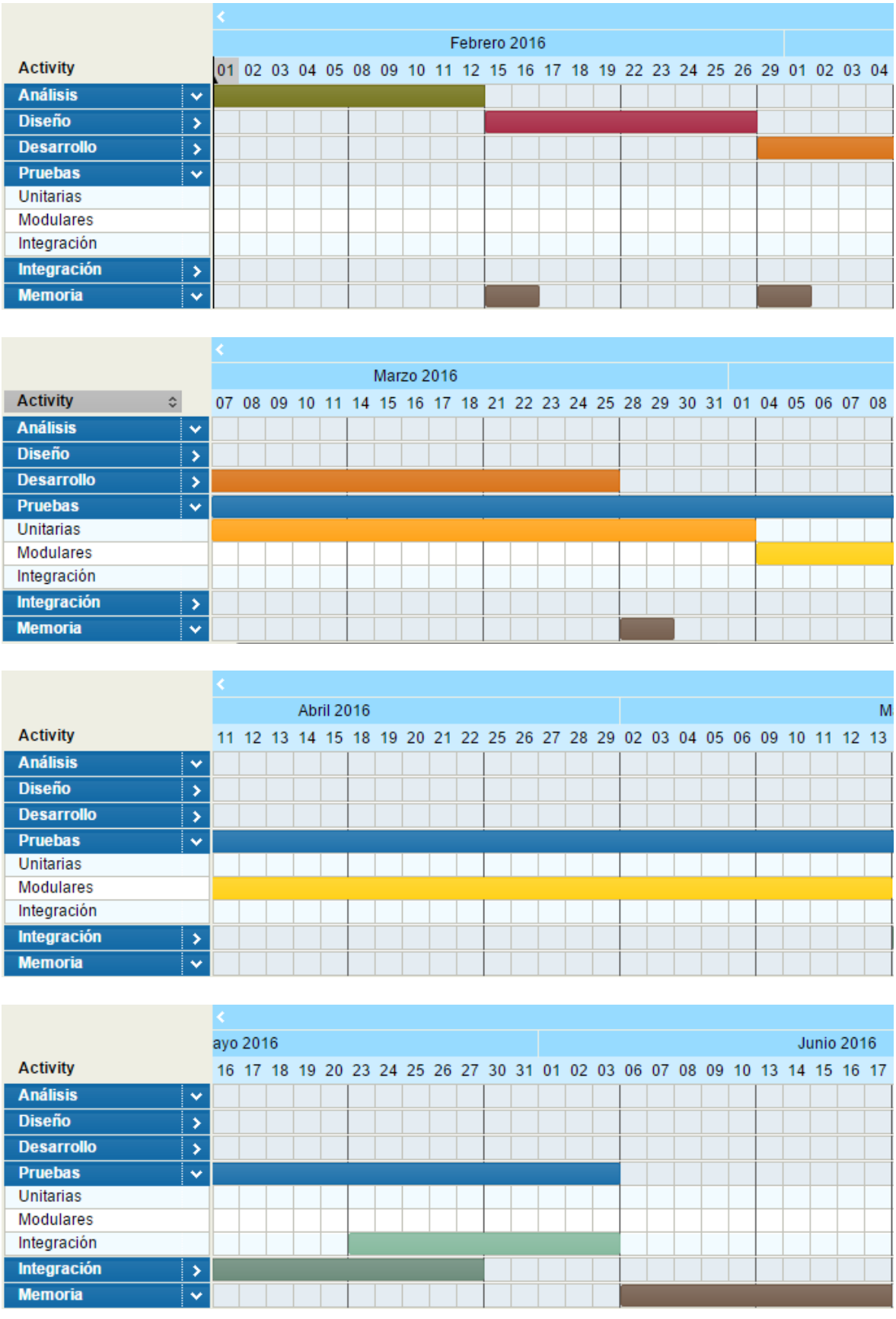


Figura 4.3-3: Diagrama de secuencia caso baja imprevista.

4.4 Diagrama de Gantt



5 Desarrollo

5.1 Introducción

Tras las fases de análisis y diseño que se han explicado en los capítulos anteriores ahora se va a detallar el proceso de desarrollo.

Para esta fase del proyecto se han creado dos máquinas virtuales idénticas, una como entorno de desarrollo y otra como entorno de producción. Estas máquinas cuentan con las siguientes características:

Hardware:

- Sistema operativo: Ubuntu
- Memoria: 4 GB
- Procesadores: 1
- Disco duro: 40 GB

Software:

- Entorno de desarrollo: NetBeans
- Servidor: Glassfish
- Base de datos: PostgreSQL

Para el control de versiones se ha usado un repositorio Git (en GitHub) junto a backups de las versiones estables en unidades externas.

5.2 Entorno de desarrollo

Como se acaba de mencionar, el desarrollo se ha realizado en una máquina virtual Linux. El marco tecnológico que se ha usado es el siguiente:

Netbeans: Es un entorno de programación especialmente indicado para el desarrollo en Java. El producto es libre, gratuito y sin restricciones de uso. Muy fácil de integrar el control de versiones para GitHub. Cuenta con el servidor GlassFish.

GlassFish: Servidor de aplicaciones de software libre.

PostgreSQL: Gestor de la base de datos.

Chrome y **Firefox**: Navegadores web con los que se ha desarrollado y hecho pruebas de compatibilidades.

PgAdmin3: Entorno gráfico para la gestión de bases de datos PostgreSQL de una manera sencilla.

5.3 Ciclo de vida

La metodología que se ha usado para el desarrollo del proyecto es una metodología en cascada. De esta manera se ha ido pasando por diferentes fases como muestra el gráfico.

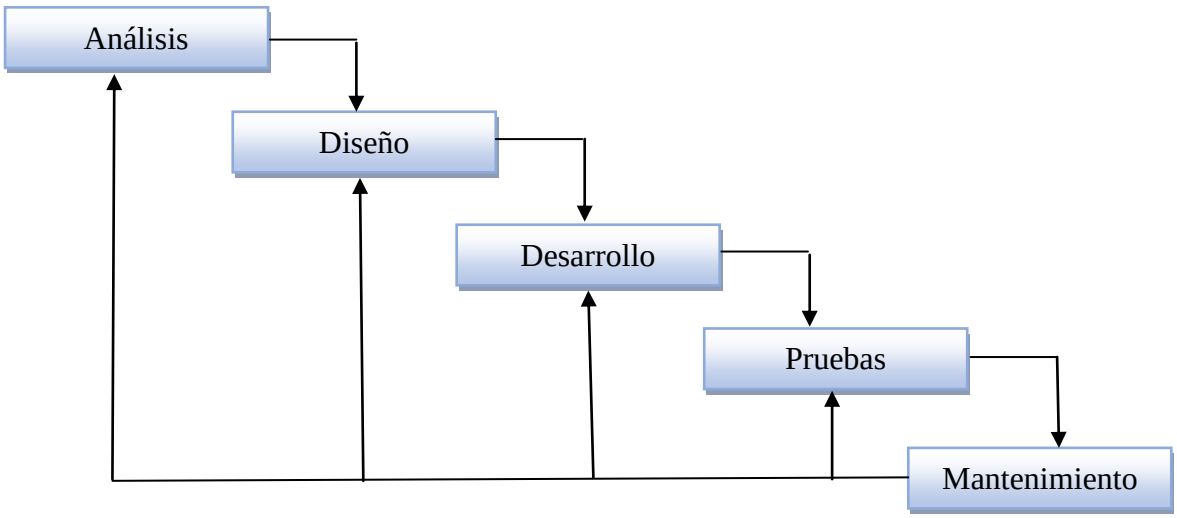


Figura 5.3-1: Ciclo de vida del proyecto software en cascada.

La fase de mantenimiento o ampliación de nuevas funcionalidades supone una vuelta a todas las fases anteriores a modo de corregir, mejorar o rediseñar la aplicación con la idea de integrar los nuevos cambios. Estos cambios pueden ser nuevos hitos o parches para corregir fallos.

5.4 Patrón MVC

Como ya se ha indicado en varias ocasiones el patrón de diseño que usa la tecnología JavaServer Faces es la de Modelo-Vista-Controlador (MVC). Este patrón se caracteriza por su separación modularizada en esos tres componentes. A continuación se va a describir el desarrollo de cada uno de ellos.

5.4.1 Modelo

El modelo JSF se define mediante Beans (Enterprise Java Beans), clases con un conjunto de atributos (denominados propiedades) y métodos getters y setters mediante los cuales se obtiene y se actualiza el contenido de esas propiedades.

Las propiedades del Bean se pueden leer y escribir desde las páginas JSF (vistas) mediante el lenguaje de expresiones EL. Cuando se llama a un Bean se hace utilizando el nombre de la propiedad. Si la expresión es de lectura se llamará al getter y este devolverá el valor y se mostrará en la vista. Si por el contrario la expresión es de escritura el método que se utilizará internamente será el set y este actualizará el contenido de la propiedad con el dato introducido en la página.

Una de las grandes virtudes de estos Beans es que soportan la invocación remota (Remote Method Invocation). RMI forma parte del entorno estándar de ejecución de Java y proporciona un mecanismo simple para la comunicación de servidores en aplicaciones distribuidas basadas exclusivamente en Java. Esto será de gran importancia en los hitos futuros de este proyecto cuando se amplíe con aplicaciones para Android ya que se podrá llamar a estos Beans y ahorra el tener que desarrollar de nuevo la lógica de negocio.

Tipos de EJB:

Entity EJB's: Manipulan copias de información residentes de un depósito de datos, en general de una base de datos. Esto permite que en caso de fallo en el Bean, no se pierda la información original.

Session EJB's: Permiten realizar cierta lógica solicitada por un cliente. Por tanto la información existirá durante el tiempo que el cliente interactúe con el Bean. Existen dos tipos:

- **Stateless (Session) EJB's**: Estos Beans sin estado son utilizados para acciones que no requieren identificar al usuario. No se modifica internamente el estado del Bean por lo que puede crearse un único Bean y estar asignado a diferentes usuarios.

- **Statefull (Session) EJB's**: Permiten mantener la sesión, identificando así al usuario. Por tanto se crea un Bean para cada cliente, el cual guarda la información de este durante el tiempo de la sesión.

Messaging EJB's: Permite recibir y publicar mensajes JMS(Java Messaging system). Este sistema funciona de forma asíncrona.

Para el desarrollo del proyecto se necesitaba poder identificar al usuario a través de sesiones y que la información se guardará, por tanto, los Beans utilizados son los de sesión sin estado (Stateless EJB's). Los Beans de sesión con estado guardan la información durante el tiempo de la sesión pero luego esta información se pierde. En este proyecto esta información debería guardarse entre sesiones por lo que se hace a través de la base de datos con Stateless EJB's.

5.4.2 Vista

Como se contó en el capítulo del "Estado del Arte", JavaServer Faces hace uso del lenguaje XHTML para presentar la información a través de las vistas. Esta tecnología es una encapsulación de código HTML en etiquetas XML. Cuenta con etiquetas propias de JSF que ayudan a la comunicación con la lógica de negocio.

Se ha contado con la librería PrimeFaces, es una extensión de JSF que cuenta con un conjunto de componentes enriquecidos lo que ayuda a lograr una interfaz más visual y amigable para el usuario. Ejemplo de definición de estas tecnologías en una vista:

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:p="http://primefaces.org/ui">
```

Figura 5.4.2-1: Ejemplo de definición de PrimeFaces

Se puede ver que para hacer uso de PrimeFaces, las etiquetas deberán ir categorizarse como 'p'. Como ejemplo se presenta el formulario de Login:

```
<p:growl id="mensajes" showDetail="true" life="2000" />
<h:form>
    <p:panel header="Login" id="loginPanel" style="width:350px">
        <h:panelGrid columns="2" cellpadding="5">
            <h:outputLabel for="username" value="Usuario:" />
            <p:inputText value="#{loginBean.username}" id="username"
                         required="true" label="username" />
            <h:outputLabel for="password" value="Clave:" />
            <p:password value="#{loginBean.password}" id="password" required="true"
                        label="password" />
            <f:facet name="footer">
                <p:commandButton id="loginButton" value="Login" action="#{loginBean.doLogin}" update=":mensajes"/>
            </f:facet>
        </h:panelGrid>
    </p:panel>
</h:form>
```

Figura 5.4.2-2: Código de login

En este código aparecen varias etiquetas haciendo uso de esta librería (growl, panel, inputText, password y commandButton). Este código junto a una guía de estilo en el archivo css da como resultado el siguiente panel de login:

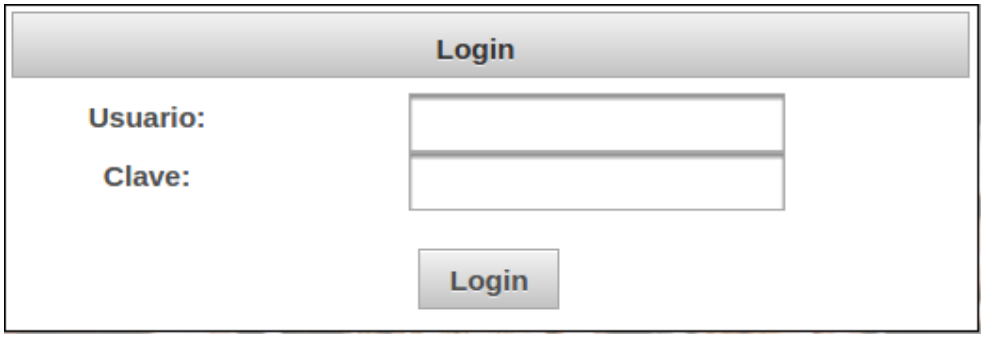


Figura 5.4.2-3: Vista del panel de login

Para la organización de los eventos uno de los elementos básicos era contar con un calendario. Este objeto ha sido un claro ejemplo de la gran ayuda que ha supuesto PrimeFaces para presentar un diseño innovador y atractivo.

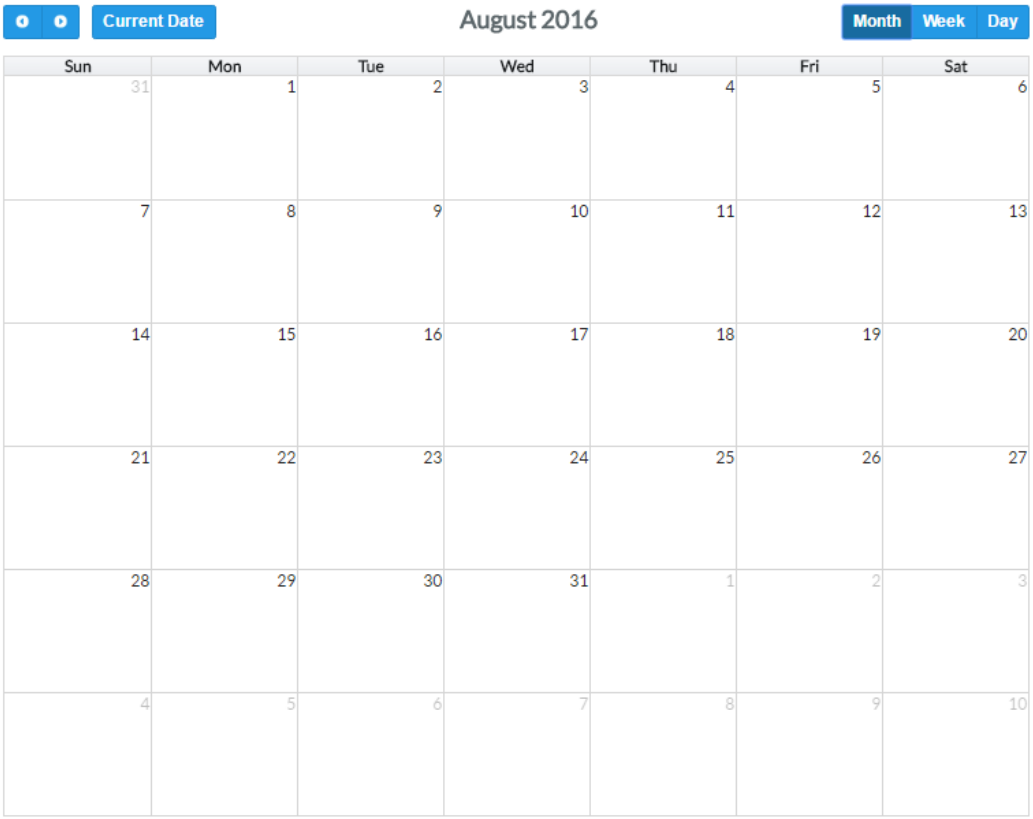


Figura 5.4.2.-4: Vista del calendario

Una parte importante de la presentación de la información es el cómo se presenta dicha información. Para el diseño de los estilos entra en acción un lenguaje muy importante: CSS. Y es que es en estos ficheros CSS donde se definen todas las directrices de estilo.

5.4.3 Controlador

Último pilar del patrón MVC. El objetivo del controlador es que las vistas puedan acceder a la lógica de negocio. Cuando el usuario realiza un evento como puede ser pulsar un botón, el controlador realiza llamadas al modelo y este actualiza la vista. Esto se hace a través de unos objetos llamado Managed Beans.

```
@ManagedBean(name = "loginController")
@SessionScoped
public class LoginController implements Serializable {

    private Login current;
    private DataModel items = null;
    @EJB
    private sesiones.LoginFacade ejbFacade;
    private PaginationHelper pagination;
    private int selectedItemIndex;

    public LoginController() {
    }
```

Figura 5.4.3.-1: Código de Login Controller

Los Manage Beans son un tipo de clase con un constructor público sin argumentos, y sus propiedades son accedidas a través de los getters y setters. Son inicializados en tiempo de ejecución.

5.5 Base de datos

Este apartado se va a tratar la estructura de la base de datos utilizada en el proyecto.

5.5.1 Modelo Entidad-Relación

El modelo Entidad/Relación o E/R es una herramienta para modelar la organización de bases de datos. Este modelo expresa entidades relevantes para un sistema de información así como sus propiedades. En él, pueden verse los grupos de datos y las relaciones que tienen con el resto de conjuntos de datos. El modelo E/R de este proyecto es el que se muestra en la siguiente figura.

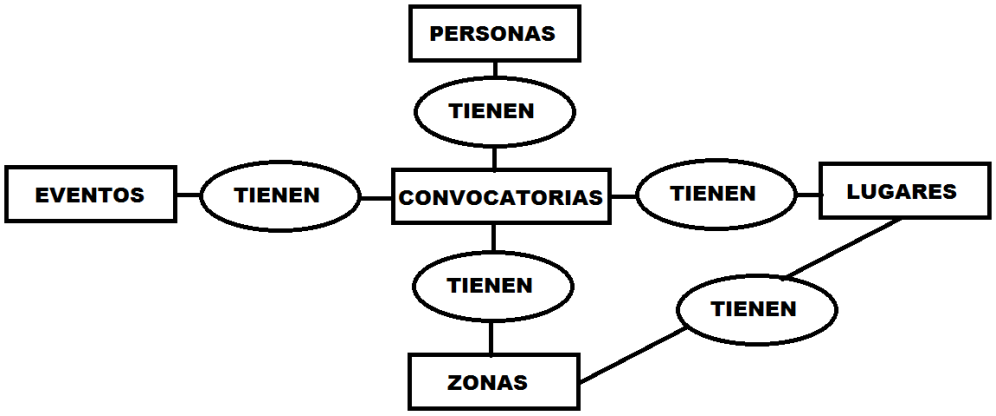


Figura 5.5.1-1: Diagrama Entidad/Relación

5.5.2 Modelo Relacional

El modelo relacional es el más utilizado actualmente para implementar BD, está basado en la lógica de predicados y en la teoría de conjuntos. Establece conexiones entre los datos permitiendo relaciones entre diferentes tablas. A continuación mostramos el modelo relacional de la base de datos creada para este proyecto.

Personas (idPersona, nombre, apellido1, apellido2, dni, email, telefono, categoria, peso, altura, colorpelo, colorojos, nSegSocial, fNacimiento, direccion, cPostal, ciudad, pais, nCuentaBanco)

Login (idRegistro, idPersona, email, password, perfil)

Eventos (idEvento, titulo, inicio, fin, observacion)

Lugares (idLugar, nombre, aforo, nTrabajadores)

Zonas (idZona, idLugar, zona)

Convocatorias (idConvocatoria, idPersona, idEvento, idLugar, idZona, disponible, asistencia)

5.5.3 Descripción de las tablas

Este apartado da una descripción de todas las tablas que forman la base de datos y una explicación de los campos que forman cada una de las tablas. Como ya se ha visto en los modelos anteriores la base de datos está compuesta de 6 tablas (Personas, Login, Eventos, Lugares, Zonas, Convocatorias). A continuación se procede a su descripción:

Personas: Tabla con la información personal de cada uno de los empleados.

- **idPersona**: Clave primaria, identifica cada una de las tuplas.
- **nombre**: Nombre del empleado.
- **apellido1**: Primer apellido de la persona.
- **apellido2**: Segundo apellido de la persona.
- **dni**: Numero del Documento Nacional de Identidad o en su defecto del pasaporte.
- **email**: Dirección de email con la que contactar a la persona.
- **teléfono**: Número de teléfono.
- **categoría**: Categoría de trabajo (azafata, protocolo, seguridad…).
- **nSegSocial**: Número de la seguridad social.
- **fNacimiento**: Fecha de nacimiento.
- **direccion**: Dirección personal del empleado.

- **cPostal**: Código postal.
- **ciudad**: Ciudad de residencia de la persona.
- **país**: País de residencia de la persona.
- **nCuentaBanco**: Número de cuenta de banco donde se realizarán los pagos.
- **peso**: Peso de la persona (Sólo para los empleados en puestos de imagen).
- **altura**: Altura de la persona (Sólo para los empleados en puestos de imagen).
- **colorPelo**: Color de pelo (Sólo para los empleados en puestos de imagen).
- **colorOjos**: Color de ojos (Sólo para los empleados en puestos de imagen).
- **tallaCalzado**: Talla de zapato.
- **tallaVestido**: Talla del vestido en el caso de las chicas.
- **tallaCamisa**: Talla de camisa
- **tallaChaqueta**: Talla de la chaqueta.
- **carnetCoche**: Si se posee carnet de coche.
- **vehiculoPropio**: Si se dispone de vehículo propio para desplazarse.
- **nivelIngles**: A1, A2, B1, B2, C1 o C2 siendo A1 el nivel básico y C2 el bilingüe.

Login: Datos de acceso a la gente registrada.

- **idUser**: Clave primaria de la tabla para los datos de acceso de una persona.
- **idPersona**: Clave secundaria que identifica a cada persona.
- **email**: La dirección de email será usada como nombre de usuario de acceso.
- **password**: Contraseña del usuario para acceder a la aplicación.
- **rol**: Indica el perfil de acceso a la aplicación. Si tiene un perfil de trabajador tendrá un acceso restringido. Si el perfil es de administrador tendrá acceso a la aplicación entera.

Eventos: Datos sobre los eventos organizados.

- **idEvento**: Clave primaria para identificar cada evento.
- **titulo**: Nombre del evento.
- **FechaInicio**: Fecha de inicio del evento.
- **HoraInicio**: Hora de inicio del evento.
- **FechaFin**: Fecha de finalización del evento.
- **HoraFin**: Hora de finalización del evento.
- **idLugar**: Clave externa que enlaza con el lugar donde se realiza el evento.
- **nTrabajadores**: Número de trabajadores necesarios para el eventos.
- **observación**: Detalles sobre el evento.

Lugares: Recintos feriales donde se organizan los eventos.

- **idLugar**: Clave primaria que identifica los recintos donde se organizan los eventos.
- **nombre**: Nombre del lugar donde se celebra un evento.
- **aforo**: Aforo del lugar.
- **direccion**: Dirección donde tendrá lugar el evento.
- **latitud**: Coordenadas para mostrarlo en un mapa.
- **longitud**: Coordenadas.

Bernabéu: Puestos de trabajo dentro del estadio.

- **idPuesto**: Clave primaria que identifica el puesto de trabajo de cada persona dentro del estadio.
- **zona**: Grupo de palcos que dependen del mismo coordinador.
- **palco**: zona VIP donde está asignado el puesto.
- **rol**: Perfil del empleado.

Convocatorias: Información a un empleado para la citación de trabajo a un evento. Esta tabla relaciona las personas, con los eventos y sus puestos de trabajo.

- **idConvocatoria**: Clave primaria que identifica cada relación Persona-Evento-Lugar-Puesto de trabajo.
- **idPersona**: Clave secundaria que relaciona a la persona convocada para trabajar en el evento.
- **idEvento**: Clave secundaria que relaciona la convocatoria con los datos del evento.
- **idZona**: Clave secundaria que relaciona la convocatoria con el puesto de trabajo para esta persona.
- **pago**: Salario que se pagará por trabajar en el evento.
- **disponible**: Campo booleano que indica si la persona confirmó asistencia para trabajar en el evento.
- **asistencia**: Indica si la persona efectivamente trabajo o causo baja después de confirmar asistencia.

5.5.4 Interacción con la base de datos

Se permite la interacción con la base de datos de diferentes maneras:

La principal de ellas es mediante formularios web en las vistas de la aplicación y es el usuario el que introduce los datos. En este caso, la aplicación será la encargada de la gestión de los datos para que los cambios sean persistentes en la base de datos. Más específicamente, los EJB (Enterprise Java Beans) se encargan de la comunicación entre la lógica de la aplicación y la base de datos. Los métodos de los EJB encargados de las consultas a la base de datos lo hacen mediante JDBC (Java Data Base Connectivity) usando un Pool de conexiones (Connection Pool).

La segunda es mediante el acceso directo a la propia base de datos. Para esta opción sólo tiene que tener acceso el administrador y es este el responsable que tras una operación, la base de datos continúe estable. Esto se puede hacer mediante dos formas: CLI (Command Line Interface) y GUI (Graphic User Interface). Para realizar operaciones en la base de datos desde la interfaz gráfica de usuario se ha usado pgAdmin3. Es un entorno visual que funciona multiplataforma (Linux, Mac

OSX, Windows). Cuando el administrador tiene que operar en la base de datos este entorno facilita enormemente la gestión.

Las acciones permitidas con la base de datos son la de inserción, borrado, modificación y consulta de datos. Según el perfil de acceso a la aplicación se podrá acceso a unos datos u otros. Por ejemplo, el empleado tan sólo puede añadir, modificar o borrar sus datos personales, sin tener acceso a los de sus compañeros. Mientras que los usuarios con perfil de administrador trabajarán con los eventos, lugares, puestos de trabajo o convocatorias.

En cuanto a las consultas ocurre lo mismo, el usuario general podrá consultar sus datos personales y los eventos a los que ha sido convocado pero no podrá realizar consultas sobre el resto de personal registrado. Los usuarios con un perfil de administrador podrán consultar a todos los datos de los empleados.

6 Integración, pruebas y resultados

Última fase del proyecto y a su vez la que más tiempo requiere. Como se puede ver en el diagrama de Gantt, las pruebas están presentes durante casi todo el periodo de desarrollo y se extiende durante varias semanas una vez acabada esta fase. Se encarga de validar el correcto funcionamiento de todo el sistema. Esto crea una espiral en la que se detectan fallos y se revisa el desarrollo, yendo y viniendo así de una fase a otra.

Se han realizado varios tipos de pruebas:

Pruebas unitarias: Se llevan a cabo durante la fase de desarrollo validando cada uno de los módulos que componen el proyecto de manera independiente. Estas pruebas son las de más bajo nivel y las más importantes. La detección un fallo en este punto puede suponer un coste muy bajo para su corrección, sin embargo, la detección de errores en niveles más altos puede elevar exponencialmente los costes. Reside aquí la gran importancia de realizar unas pruebas unitarias correctas.

Para realizar estas pruebas se ha usado JUnit. Es un conjunto de bibliotecas que permite realizar la ejecución de clases Java de manera controlada y así poder testear el correcto funcionamiento de cada uno de los métodos. Se define una pila de pruebas para cada método en la que se declara los inputs y los outputs esperados y se ejecutan todas de manera secuencial. Una vez termina se pueden ver los resultados de manera muy visual.

Pruebas de integración: Una vez se ha comprobado que cada módulo por separado funciona como se espera, se han realizado pruebas centradas en la comunicación entre los componentes del sistema.

En este punto se da por validado el sistema en el entorno de desarrollo. Ahora, se migra el proyecto al entorno de producción. Esta máquina de producción tiene las mismas características que el de desarrollo pero existen otras aplicaciones en ejecución. Hay que validar que todas estas aplicaciones continúan su ejecución de manera correcta sin entrar en conflicto unos sistemas con otros. Un caso típico, es que algún proceso consuma demasiados recursos de la máquina, lo cual repercuta en el rendimiento de las demás aplicaciones.

Pruebas de validación: Últimas pruebas para validar el sistema en el entorno de producción. Estas pruebas, a parte de una correcta ejecución, tienen en cuenta otros factores como los tiempos de respuesta. Aunque el rendimiento no es punto crítico de esta aplicación, sí es necesario que la experiencia de navegación del usuario cumpla con unos mínimos.

7 Conclusiones y trabajo futuro

7.1 Conclusiones

La realización de este trabajo ha supuesto una experiencia enriquecedora en el campo de la ingeniería del Software y de los Sistemas Informáticos. Es un trabajo que ha permitido realizar un proyecto de Software de principio a fin en el que se han unido los conocimientos de varias asignaturas estudiadas durante estos años. Para hacer posible este proyecto, primero se ha localizado un problema real en la gestión profesional de eventos de gran envergadura y se ha imaginado una posible solución. Durante los meses que ha durado el trabajo se ha ido dando forma a esta idea hasta convertirlo en un producto real y usable que cumple con la funcionalidad para la que se diseñó, mejorar la gestión interna de este tipo de empresas.

Durante este camino se ha aprendido multitud de tecnologías, se ha probado y estudiado diferentes frameworks hasta que se ha visto cual era el óptimo para el desarrollo de este proyecto. Lo mismo ha ocurrido con los gestores de bases de datos, los servidores web o los entornos de desarrollo. Esto ha supuesto sobretodo mucha lectura sobre todos estos sistemas.

Una vez se tenía definido el entorno completo de desarrollo y hasta lograr una correcta configuración de todas las tecnologías, se han tenido que superar múltiples problemas y errores de todo tipo. Esto ha hecho más complicado el camino, pero a su vez, ha ayudado a lograr un gran conocimiento sobre el sistema que se administra.

Gracias a lo mucho que se ha aprendido y al interés que ha generado se va a continuar con el proyecto de manera que pueda ser integrado en una empresa real. En el siguiente apartado se explican las diferentes líneas de trabajo que se han definido.

7.2 Trabajo futuro

Este proyecto es de mayor ambición que el que se ha presentado en este punto. Se va a seguir trabajando en él para lograr varias mejoras de cara a poder ser utilizada en un entorno profesional. Existen 3 líneas abiertas en las que se va a seguir trabajando:

El primer punto sería migrar el proyecto a un servidor en Internet ya que actualmente el servidor trabaja en localhost. Habría que tener accesibilidad desde Internet para que cualquiera pudiera interactuar con la aplicación. Una vez hecha la migración habría que volver hacer unas pruebas de validación para comprobar que todo continua funcionando correctamente. También habría que plantearse ciertas medidas de seguridad en el servidor como que puertos tiene abiertos para poner ciertas barreras para impedir accesos de manera no permitida. Por último, habría que monitorizar la salud de la máquina para poder generar alertas en caso de fallo como puede ser la conectividad, el % de la CPU o la memoria.

El segundo punto en el que se va a trabajar es la mejora de funcionalidades. Una vez ya se tiene la primera versión de la aplicación, que es la que se presenta en este trabajo, se irá mejorando su funcionalidad. Cada uno de estos nuevos hitos serán proyectos independientes de menor envergadura pero con un ciclo de vida idéntico al del proyecto completo. Una opción de importante es la de poder generar reportes y descargarlos para tener archivos offline y no depender siempre de la aplicación.

El tercer punto de mejora es la posibilidad de interactuar con usuarios a través de plataformas móviles y tablets Android. Esto supondrá un gran diferencial con el resto de aplicaciones de gestión ya que permitirá a todos los coordinadores poder usar la aplicación durante el propio evento de manera móvil y desde cualquier punto del recinto donde tiene lugar, sin depender de un computador tradicional.

Referencias

[1] Tutorial JavaServer Faces, http://docs.oracle.com/javaee/6/tutorial/doc/bnaph.html
[2] Tutorial JavaServer Faces, http://www.coreservlets.com/JSF-Tutorial/jsf2/#Whirlwind
[3] JavaServer Faces 2.0, The Complete Reference, Ed Burns, Chris Schalk, McGraw-Hill, 2009
[4] Tutorial Ajax with JSF, http://docs.oracle.com/javaee/6/tutorial/doc/gkiow.html
[5] PrimeFaces, http://www.primefaces.org/
[6] Comparativa de Frameworks web, http://static1.1.sqspcdn.com/static/f/923743/15025206/1320739503647/frameworks_web.pdf?token=0p0jLXicjEHOLxPkNFSBDnZXMYo%3D
[7] La guía definitiva de Django, Adrian Holovaty, Anaya Multimedia, 2009
[8] PostgreSQL y MySQL, https://2ndquadrant.com/es/postgresql/postgresql-vs-mysql/
[9] PostgreSQL y MySQL, https://www.wikivs.com/wiki/MySQL_vs_PostgreSQL
[10] Tendencias, https://www.google.es/trends/
[11] Esquema relacional, http://www.web2py.com/sqldesigner
[12] MVC en JavaServer Faces, http://www.jtech.ua.es/j2ee/publico/jsf-2012-13/sesion02-apuntes.html
[13] Sistema de Gestión de Bases de Datos, https://es.wikipedia.org/wiki/Sistema_de_Gesti%C3%B3n_de_Bases_de_Datos
[14] Symfony, https://es.wikipedia.org/wiki/Symfony
[15] Ruby on Rails, https://es.wikipedia.org/wiki/Ruby_on_Rails
[16] ASP.NET, https://es.wikipedia.org/wiki/ASP.NET
[17] MongoDB, https://es.wikipedia.org/wiki/MongoDB
[18] MySQL, https://es.wikipedia.org/wiki/MySQL
[19] Oracle Database, https://es.wikipedia.org/wiki/Oracle_Database
[20] PostgreSQL, https://es.wikipedia.org/wiki/PostgreSQL
[21] Microsoft SQL Server, https://es.wikipedia.org/wiki/Microsoft_SQL_Server#cite_note-2
[22] SQLite, https://es.wikipedia.org/wiki/SQLite
[23] Enterprise JavaBeans, https://es.wikipedia.org/wiki/Enterprise_JavaBeans

Glosario

API	Application Programming Interface.
Bean	Componente con una funcionalidad lógica.
BD	Base de Datos.
CSS	Cascading Style Sheets. Hoja de estilos.
Framework	Conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar.
HTML	HyperText Markup Language.
JDBC	Java Data Base Connectivity. API de conexión a la base de datos.
JSF	JavaServer Faces. Tecnología para el desarrollo de aplicaciones web.
MVC	Modelo-Vista-Controlador. Patrón de diseño.
PrimeFaces	Conjunto de librerías para el desarrollo de las vistas en JSF.
SQL	Structured Query Language. Lenguaje para operar en bases de datos.